**CS-171 Checkers Final AI Report**

**Team name**: WALLE

**Member #1 (name and NetID):** Mamta Jayesbhai Shah (mamtajs)

**Member #2 (name and NetID):** Yixing Wang (yixingw)

**I. In about 1/2 page of text, describe what you did to make your Final AI agent “smart.”**

 For every getMove() call we first check for all the valid moves, if putting opponent’s mark at one of those positions makes the opponent to win if we find such a position we immediately move to that position. If not, then we need to choose the position to move to wisely by a heuristic which is different cases where gravity is on and off as follows:

**For Gravity ON:** We understood that the possible move for us are restricted to the number of columns. So, we maintained a list of possible value of row for each column which gives us possible moves (col, row) to evaluate. Maximum moves that we need to evaluate and compare are less than or equal to number of columns in the board. We evaluate these positions by increasing their desirability based on the following rule:

1. If we are blocking opponent’s mark on both left and right of the cell increase desirability of position by 2. If we only block on left or right increase by 1.
2. Opposite to above point if have our mark on both left and right and can connect the two by choosing this cell as our move then increase desirability by 4. And if only either of left and right is our mark then increase it by 3.
3. Similar to above points 1 and 2, we now consider diagonals from top-right to bottom-left and top-left to bottom-right. If we find that we can block both the marks diagonal to a position for opponent, then we increase the desirability of the move by 3 and if we can connect those for ourselves, we increase desirability by 5.
4. If only one of the top-left, bottom-right, top-right and bottom-left marks are blocked for opponent then we increase the desirability by 2 and if we can connect any one for ourselves, we increase desirability by 4.

The intuition for this heuristic function is to consider horizontal line, and both diagonals with respect to a particular possible move/cell. We want to increase our chances to win more than blocking the opponent and hence we the values increase as mentioned above.

**For Gravity OFF:** With gravity on, we need to consider every cell as potential candidates for the next move. For each cell, we evaluate its desirability according to its all eight adjacent neighbors. Given that the board size is relatively small, it’s possible to compute for every cell within fairly short time. We evaluate these cells by increasing their desirability based on the similar rule as mentioned above, but we are playing more inaggressively, which will lead to more ties than winning. The evaluation function is based on the following rules:

1) If we are blocking opponent’s mark on both left and right of the cell increase desirability of position by 4. If we only block on left or right increase by 3.

2) Opposite to above point if have our mark on both left and right and can connect the two by choosing this cell as our move then increase desirability by 2. And if only either of left and right is our mark then increase it by 1.

3)Similar to above points 1 and 2, we now consider top and bottom cells for the potential move. The heuristic values are the same as left-right ones.

4) Similar to above points 1 and 2, we now consider diagonals from top-right to bottom-left and top-left to bottom-right. If we find that we can block both the marks diagonal to a position for opponent, then we increase the desirability of the move by 4 and if we can connect those for ourselves, we increase desirability by 2.

5) If only one of the top-left, bottom-right, top-right and bottom-left marks are blocked for opponent then we increase the desirability by 3 and if we can connect any one for ourselves, we increase desirability by 1.

**II. In about 1/4 page of text, describe problems you encountered and how you solved them.**

**** We had to struggle setting up the python environment during first project, we realized it was due to incompatible version of python and so we installed the exact version of python recommended which solved this problem.

We also came across the difficulty of coming up with a good way to evaluate a position better by setting the weights/desirability of a position in a given state. We solved this by taking inspiration from the way we play tic tac toe and experimenting with different weights and comparing the results for all the tests using self-implemented test script.

**III. In about 1/4 page of text, provide suggestions for improving this project.**

 We don’t see any more improvements needed for this project, it is in great shape already given the clarity in tasks and how to approach the problem as well as the fun leaderboards, we just want to thank for all that.